REMARKS

This Preliminary Amendment is filed in response to the Final Office Action dated January 16, 2008, along with a Request for Continued Examination and the appropriate fees. All objections and rejections are respectfully traversed.

Claims 1-11, 13-26, and 28-49 are currently pending.

Claims 1, 11, 20-22, 32, 38, and 48 have been amended to better claim the invention.

Claim 49 has been added to better claim the invention.

Request for Interview

The Applicant respectfully requests a telephonic interview with the Examiner after the Examiner has had an opportunity to consider this Amendment, but before the issuance of the next Office Action. The Applicant may be reached at 617-951-2500.

Claim Rejections – 35 USC §103

At paragraph 3 of the Office Action, claims 11 and 13-19 were rejected under 35 U.S.C. §103(a) as being unpatentable over Hitz et al. U.S. Patent No. 5,819,292, issued on October 6, 1998 (hereinafter "Hitz"), in view of Ganesh et al., U.S. Patent No. 6,197,377, issued on February 20, 2001 (hereinafter "Ganesh"). Additionally, at paragraph 4 of the Office Action, claims 1-10, 20-26, and 28-48 were rejected under 35 U.S.C. §103(a) as being unpatentable over Hitz, in view of Marion et al., U.S. Patent Application Publication No. 2003/0163661, published on August 28, 2003 (hereinafter "Marion"), and in further view of Ganesh.

The present invention, as set out in representative claim 1, comprises in part:

> 1. A method for detecting leaked buffer writes between a first consistency point and a second consistency point, the method comprising:

receiving a write operation, the write operation identifying a file for the write operation;

determining that a volume storing the file has buffer leakage detection activated;

creating a data buffer associated with the write operation; and

in response to determining the volume has buffer leakage detection activated, writing a buffer check control structure to a raw data buffer associated with the data buffer, the buffer check control structure including one or more uniquely identifying numbers referred to as magic numbers and a consistency point number, the magic numbers to uniquely identify the raw data buffer as a labeled buffer check control structure and to indicate that the data structure needs to be checked for leakage.

Hitz discloses a method for maintaining consistent states of a file system. In particular, incore WAFL Inodes contain information stemming from an on-disk inode, a WAFL buffer data structure, and buffer pointers. Specifically, this incore inode contains incore information pertaining to the buffer structure, where the incore information is made up of a dirty flag, an in-consistency point (IN_CP) flag and pointers for a linked list. The dirty flag represents an inode that has been modified or references buffers that have changed. The IN_CP flag, on the other hand, is used to mark an inode as being in a consistency point. Therefore, these flags are used to indicate when the inode is in a consistency point and must be cleared and written to a disk during its consistency point.

Marion discloses a memory divided into memory array slots. Each slot has a flag which indicates whether the slot is allocated or deallocated. Specifically, Marion generates a memory leak report which is processed when a memory leak flag is received from a user. In Marion, the user sets the memory leak flag to notify the system that it should be tracking the memory allocations and deallocations. Subsequently, a determination is made as to whether the user chooses to track the memory leaks using the memory leak flag. If the user chooses to track the memory leaks, memory is allocated for an OS-memory array allowing the OS-memory array to begin tracking the memory leaks. At this point, a determination is made as to whether the system should generate a report.

When the user requests that a report be generated, a report is generated using memory leak information in the OS-memory array. Finally, this memory leak information is stored in a non-volatile storage area.

Ganesh discloses using a snapshot and a transaction summary to determine if a particular data block is invalid and therefore can be overwritten. More specifically, by way of its background, Ganesh discloses evaluating changes made to different versions of data blocks to determine whether any of those versions can be used during a read request. One approach Ganesh discusses is maintaining, for each version of a data block, a list that specifies any transactions that may have modified that version of the data block (i.e., a transaction log). When a transaction comes in, the list is examined to determine if any of the various versions can be used by the transaction. For instance, if the list for a particular version of the data block indicates that there have been changes made by a transaction that have not been committed, then that version of the data block cannot be used by the transaction. Furthermore, if the list of records for a particular version indicates that the version does not include changes that must be seen by a transaction then that version cannot be used by a new transaction. This list of records is generally in the form of an index that contains information about transactions that have been updated. Particularly, each entry in the list corresponds to an update which has an index number, a transaction ID associated with the transaction that made the change, a status flag, a snapshot number for the transaction that updated the data block, and other various information. The index number and transaction ID point to each particular entry and transaction respectively associated with a version so that they can be identified by the system (i.e., they act as reference tags). The status flag merely indicates whether the transaction is currently active or not to prevent inspection of all the transaction list entries. A snapshot number identifies a randomly assigned number for the transaction that updated the data block (Ganesh has no further discussion of snapshot numbers).

Applicant respectfully urges that neither Hitz, Marion, nor Ganesh disclose Applicant's claimed novel ***in response to determining the volume has buffer leakage***

detection activated, writing a buffer check control structure to a raw data buffer associated with the data buffer, the buffer check control structure including one or more uniquely identifying numbers referred to as magic numbers and a consistency point number, the magic numbers to uniquely identify the raw data buffer as a labeled buffer check control structure and to indicate that the data structure needs to be checked for leakage.

Applicant's claimed invention is directed toward a technique for detecting leaked buffer write operations across file system consistency points. In particular, the system receives a write operation that identifies a file to be stored. The system then determines whether a volume storing the file has buffer leakage detection activated, and if so, writes a buffer check control structure to a an associated data buffer. The buffer check control structure includes one or more uniquely identifying numbers (referred to as "magic numbers") and a consistency point number. In particular, magic numbers distinctly and uniquely identify the buffer check control structure, and are written to the raw data structure in order to distinguish between a labeled buffer check control structure from non-labeled areas, e.g., thereby preventing false detections or positives. For example, if two magic numbers are embedded in the buffer check control structure of the labeled buffer check control structure (e.g., two 32 bit numbers), each write has significantly less chance of randomly selecting the bits used for the magic numbers, so non-labeled raw data is excluded (e.g., in the case of two 32 bit number there would be a 2^64 chance of a false positive, unlike a simple flag, which may be easily corrupted). That is, there is less of a chance of incorrectly identifying the a buffer check control structure is in place. Furthermore, if a buffer check control structure is detected, the system checks to see if a current consistency point number is the same as a next consistency point number. If both consistency point numbers match, there has been buffer leakage. In other words, in Applicant's claimed invention, the magic numbers are used to confirm that a buffer check control structure is present, and the combination of the magic numbers and the consistency point number are used to determine whether buffer leakage has occurred.

As noted above, Ganesh merely discloses a list of records for a particular version of a data block which indicates that the version does not include changes that must be seen by a transaction. Ganesh does not disclose the use of magic numbers, but only index numbers which point to each particular entry and transaction respectively associated with a version so that they can be identified by the system (i.e., they act as reference tags, pointing to a particular entry in the list). Applicant's claimed magic numbers do not index into a list, but rather uniquely identify the presence of a buffer check control structure. Furthermore, as discussed above, a flag like in Marion is not the same as the applicant's magic numbers because the buffer control structure may not be correct in the first place (i.e., may be corrupted). That is, a simple flag (e.g., a value of '1' or '0') is not specifically detectable (i.e., uniquely identifiable or differentiated), and do not perform the same function as Applicant's magic numbers. Additionally, the Examiner agrees that neither Hitz nor Marion disclose the use of magic numbers, and Marion merely discloses a memory leak report which is processed when a memory leak flag is received from a user.

Applicant respectfully urges that Hitz, Marion, or Ganesh, either taken either singly or in any combination, are legally insufficient to render the presently claimed invention obvious under 35 U.S.C 103(a) because of the absence in each of the cited patents of Applicant's claimed novel ***in response to determining the volume has buffer leakage detection activated, writing a buffer check control structure to a raw data buffer associated with the data buffer, the buffer check control structure including one or more uniquely identifying numbers referred to as magic numbers and a consistency point number, the magic numbers to uniquely identify the raw data buffer as a labeled buffer check control structure and to indicate that the data structure needs to be checked for leakage***.

Conclusion

All independent claims are believed to be in condition for allowance.

All dependent claims are believed to be dependent from allowable independent claims.

Favorable action is respectfully solicited.

Please charge any additional fee occasioned by this paper to our Deposit Account No. 03-1237.

Respectfully submitted,

/James M. Behmke/
James M. Behmke
Reg. No. 51,448
CESARI AND MCKENNA, LLP
88 Black Falcon Avenue
Boston, MA 02210-2414
(617) 951-2500